NTNU

Kunnskap for en bedre verden

Institutt for matematiske fag

Eksamensoppgave i
ST2304 Statistisk modellering for biologer/bioteknologer

Faglig kontakt under eksamen: Bob O'Hara

Tlf: 915 54 416

Eksamensdato: 8. august 2018

Eksamenstid (fra–til): 09:00–13:00

Hjelpemiddelkode/Tillatte hjelpemidler: C: Ett gult A4-ark med dine egne håndskrevne notater (stemplet av Institutt for matematiske fag), Tabeller og formler i statistikk (Tapir forlag, Fagbokforlaget), Matematiske formelsamling (K. Rottmann). Bestemt kalkulator.

Annen informasjon:
Alle svar må begrunnes, og besvarelsen skal inneholde relevante mellomregninger. Hjelpesider for noen R-funksjoner er vedlagt.

Målform/språk: bokmål

Antall sider: 9

Antall sider vedlegg: 3

Kontrollert av:

Informasjon om trykking av eksamensoppgave	
Originalen er:	
1-sidig ☐	**2-sidig** ☒
sort/hvit ☒	**farger** ☐
skal ha flervalgskjema ☐	

Dato Sign

Oppgave 1 Olympiske rekorder

For noen år siden publiserte noen forskere en analyse av vinnertider for 100 m sprint i de olympiske leker (OL). De fremsatte en påstand om at vinnertiden i kvinneklassen ville være raskere enn i herreklassen i OL i 2156. Dataene ser du i figur 1.

Figur 1: Vinnertider for 100 m for menn og kvinner i OL.

Vinnertiden for kvinner ble modellert med lineær regresjon, med år som kovariat og vinnertid (målt i sekunder) som respons, se figur 2.

a) Skriv ned modellen for vinnertiden til kvinner i matematisk notasjon.

b) Hvilke antagelser må vi gjøre når vi bruker denne modellen?

c) Hva er ukjente parametere i denne modellen?

d) Hva er predikert endring i tiden fra et OL til det neste? Merk: tid mellom OL er 4 år.

En tilsvarende modell for endring i vinnertider for menn ble tilpasset (figur 3).

e) Skriv ned estimatene for regresjonsparametrene fra modellene for menn og for kvinner, og kommenter verdiene.

```
> mod.women <- lm(WinningTime ~ Year, data=Women100m)
> summary(mod.women)

Call:
lm(formula = WinningTime ~ Year, data = Women100m)

Residuals:
     Min       1Q   Median       3Q      Max
-0.37579 -0.08460  0.00929  0.08285  0.32234

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.347049   4.284251   10.35 1.70e-08 ***
Year        -0.016822   0.002176   -7.73 8.63e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2104 on 16 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared:  0.7888,        Adjusted R-squared:  0.7756
F-statistic: 59.76 on 1 and 16 DF,  p-value: 8.626e-07
```

Figur 2: R-kode og resultater av tilpasning av regresjon til vinnertider for 100 m for kvinner i OL.

```
> mod.men <- lm(WinningTime ~ Year, data=Men100m)
> summary(mod.men)

Call:
lm(formula = WinningTime ~ Year, data = Men100m)

Residuals:
      Min        1Q    Median        3Q       Max
-0.263708 -0.052702  0.007381  0.080048  0.214559

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 31.8264525  1.6796428   18.95 4.11e-15 ***
Year        -0.0110056  0.0008593  -12.81 1.13e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1347 on 22 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.8817,        Adjusted R-squared:  0.8764
F-statistic:   164 on 1 and 22 DF,  p-value: 1.128e-11
```

Figur 3: R-kode og resultater av tilpasning av regresjon til vinnertider for 100 m for menn i OL.

```
> mod.full <- lm(WinningTime ~ Sex*Year, data=Olymp100m)
> anova(mod.full)

Analysis of Variance Table

Response: WinningTime
          Df Sum Sq Mean Sq  F value    Pr(>F)
Sex        1 8.5566  8.5566 293.5207 < 2.2e-16 ***
Year       1 5.3937  5.3937 185.0198 3.507e-16 ***
Sex:Year   1 0.2292  0.2292   7.8615  0.007911 **
Residuals 38 1.1078  0.0292
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figur 4: R-kode og resultater fra variansanalyse for regresjon av vinnertider for 100 m for menn og kvinner i OL.

Hvis vi ønsker å teste om vinnertidene til kvinner endrer seg med en annen rate enn vinnertidene til menn, må vi tilpasse en ny modell til datasettet bestående av vinnertider for både menn og kvinner. Koden for å tilpasse denne modellen og en variansanalysetabell er vist i figur 4: Kjønn (sex) er en faktor som koder om øvelsen var for menn (nivået for skjæringspunktet) eller kvinner.

f) Hvilken av testene som rapporteres i variansanalysetabellen (ANOVA-tabellen) tester om vinnertidene til kvinner endres med en annen rate enn for menn? Hvilken fordeling brukes i denne testen?

g) Vil du konkludere med at vinnertiden for kvinner endrer seg med en annen rate enn vinnertiden for menn? Hva er testobservatoren og p-verdien?

Et sammendrag (summary) av regresjonsmodellen finnes i figur 5.

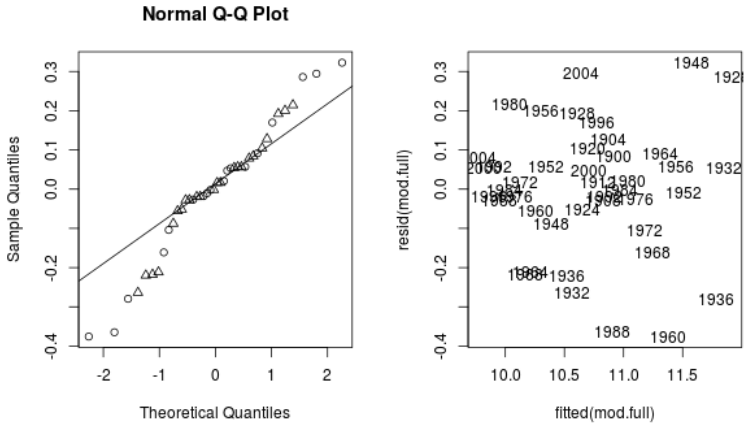
h) Hva er ligningene for vinnertidene til menn og vinnertidene til kvinner?

i) Hvor mye endres vinnertidene til kvinnene sammenlignet med vinnertidene til mennene?

j) Basert på en slik modell for vinnertider foreslo en statistiker at 100 m-sprinten i OL i 2636 vil være «veldig interessant». Hvorfor?

```
> summary(mod.full)

Call:
lm(formula = WinningTime ~ Sex * Year, data = Olymp100m)

Residuals:
     Min       1Q   Median       3Q      Max
-0.37579 -0.05460  0.00738  0.08276  0.32234

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   31.826453   2.128910  14.950  < 2e-16 ***
SexWomen      12.520596   4.076141   3.072  0.00392 **
Year          -0.011006   0.001089 -10.104 2.56e-12 ***
SexWomen:Year -0.005817   0.002074  -2.804  0.00791 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1707 on 38 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.9275,        Adjusted R-squared:  0.9218
F-statistic: 162.1 on 3 and 38 DF,  p-value: < 2.2e-16
```

Figur 5: R-kode og resultater av tilpasning av regresjon til vinnertider for 100 m for menn og kvinner i OL.



Figur 6: Plott for å sjekke modelltilpassing for vinnartider for 100 m for menn (svart) og kvinner (grått) i OL.

k) Synes du at dette er en fornuftig modell? Forklar (kort).

Vi undersøker modelltilpasningen i figur 6. Normalplott vises sammen med et plott av residualer mot tilpassede verdier.

l) Ser residualene normalfordelte ut? Begrunn svaret ditt (en til to setninger).

m) Vil du basert på residualplottene si at modellantagelsene er oppfylt?

n) Hvis du vil teste om det er en ikke-lineær effekt over år, hvordan vil du gjøre det?

Oppgave 2 Spurv i Nord-Amerika

The North American Breeding Bird Survey er en studie som utføres i Nord-Amerika med mål å estimere bestanden av fugler. Fugletittere besøker ulike steder i Nord-Amerika og teller antall fugler de observerer ved et bestemt tidspunkt. Dataene kan brukes til å undersøke en rekke spørsmål. Her skal vi se på effekten av klima på bestand av spurv.

Datasettet består av 1714 observasjoner fra ulike målepunkter i Nord-Amerika. For hvert målepunkt er antallet fugler observert (Count). Gjennomsnittstemperatur (temp.mean.sc) og total nedbør (perc.mean.sc) er hentet fra en standard database. Både verdier for temperatur og nedbør er sentrert og skalert, slik at en temperaturverdi på 0.3 betyr at temperaturen er 0.3 standardavvik over gjennomsnittet i Nord-Amerika.

Først ble en generalisert lineær modell (poissonregresjon) tilpasset til antallet spurv som respons, med gjennomsnittstemperatur og total nedbør som forklaringsvariabler (figur 7).

a) Skriv ned modellantagelsene og en ligning for det forventede antall spurver observert i et målepunkt.

b) Hva er den estimerte koeffisienten for effekten av total nedbør (prec.mean.sc)? Og, hva er et 95% konfidensintervall for koeffisienten?

c) Er det tegn på over- eller underdispersjon i data?

d) Hva er det predikerte antallet spurv i følgende målepunkter:

1. et målepunkt nært Seattle der gjennomsnittstemperaturen er 0.3 standardavvik under gjennomsnittet, og total nedbør er 0.3 standardavvik over gjennomsnittet
2. et målepunkt litt vest for Seattle, i den tempererte regnskogen, der gjennomsnittstemperaturen er 0.4 standardavvik under gjennomsnittet, men total nedbør er 5.8 standardavvik over gjennomsnittet

Fra økologisk teori vil vi forvente at det finnes en optimal temperatur og nedbør for spurver, og at bestanden går ned når forholdene er langt fra dette optimumet. Vi kan modellere dette ved å bruke en kvadratisk kurve. En modell av denne typen ble tilpasset (figur 8).

e) Støtter dataene denne økologiske teorien? Hvilke(n) parameterverdi(er) forteller oss om dette?

f) Hva er predikert antall spurv for de to målepunktene nevnt over?

g) Kommenter kort på forskjellene i prediksjonene fra den lineære til den kvadratiske modellen.

```
> mod.sparrows <- glm(Count ~ prec.mean.sc + temp.mean.sc,
                family=poisson, data=Data)
> summary(mod.sparrows)

Call:
glm(formula = Count ~ prec.mean.sc + temp.mean.sc, family = poisson,
    data = Data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0784  -2.0569  -0.9564   0.9732   8.9266

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.854271   0.009564 193.889  < 2e-16 ***
prec.mean.sc  0.040401   0.010177   3.970 7.19e-05 ***
temp.mean.sc -0.045191   0.010200  -4.431 9.39e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 8452.4  on 1713  degrees of freedom
Residual deviance: 8425.5  on 1711  degrees of freedom
AIC: 14112

Number of Fisher Scoring iterations: 5
```

Figur 7: R-kode og sammendrag (summary) fra poissonregresjon for spurvebestand i Nord-Amerika.

```
> mod.sparrows2 <- glm(stoptotal ~ prec.mean.sc + temp.mean.sc +
>               I(prec.mean.sc^2) + I(temp.mean.sc^2),
>               family=poisson, data=Data)
> disp <- mod.sparrows2$deviance/mod.sparrows2$df.residual
> summary(mod.sparrows2, dispersion=disp)


Call:
glm(formula = stoptotal ~ prec.mean.sc + temp.mean.sc + I(prec.mean.sc^2) +
    I(temp.mean.sc^2), family = poisson, data = Data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1822  -1.9344  -0.8248   0.9953  10.7981

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        2.09946    0.03028  69.332  < 2e-16 ***
prec.mean.sc      -0.00121    0.02534  -0.048    0.962
temp.mean.sc       0.03841    0.02605   1.475    0.140
I(prec.mean.sc^2) -0.10552    0.02085  -5.061 4.17e-07 ***
I(temp.mean.sc^2) -0.16822    0.02181  -7.713 1.23e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 4.614599)

    Null deviance: 8452.4  on 1713  degrees of freedom
Residual deviance: 7886.4  on 1709  degrees of freedom
AIC: 13577

Number of Fisher Scoring iterations: 5
```

Figur 8: R-kode og sammendrag (summary) fra poissonregresjon for spurvebestand i Nord-Amerika.

`lm` *Fitting Linear Models*

Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Arguments

`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted least squares is used with weights `weights` (that is, minimizing `sum(w*e^2)`); otherwise ordinary least squares is used. See also 'Details',
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`method`	the method to be used; for fitting, currently only `method = "qr"` is supported; `method = "model.frame"` returns the model frame (the same as with `model = TRUE`, see below).
`model, x, y, qr`	logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
`singular.ok`	logical. If `FALSE` (the default in S but not in R) a singular fit is an error.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See `model.offset`.
`...`	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Models for `lm` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first + second + first:second`.

If the formula includes an `offset`, this is evaluated and subtracted from the response.

If `response` is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See `model.matrix` for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms` object as the formula (see `aov` and `demo(glm.vr)` for an example).

A formula has an implied intercept term. To remove this use either `y ~ x - 1` or `y ~ 0 + x`. See `formula` for more details of allowed formulae.

Non-NULL `weights` can be used to indicate that different observations have different variances (with the values in `weights` being inversely proportional to the variances); or equivalently, when the elements of `weights` are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations (including the case that there are w_i observations equal to y_i and the data have been summarized).

`lm` calls the lower level functions `lm.fit`, etc, see below, for the actual numerical computations. For programming only, you may consider doing likewise.

All of `weights`, `subset` and `offset` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

Value

`lm` returns an object of `class` `"lm"` or for multiple responses of class `c("mlm", "lm")`.

The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

An object of class `"lm"` is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`residuals`	the residuals, that is response minus fitted values.
`fitted.values`	the fitted mean values.
`rank`	the numeric rank of the fitted linear model.
`weights`	(only for weighted fits) the specified weights.
`df.residual`	the residual degrees of freedom.
`call`	the matched call.
`terms`	the `terms` object used.
`contrasts`	(only where relevant) the contrasts used.
`xlevels`	(only where relevant) a record of the levels of the factors used in fitting.
`offset`	the offset used (missing if none were used).
`y`	if requested, the response used.
`x`	if requested, the model matrix used.
`model`	if requested (the default), the model frame used.
`na.action`	(where relevant) information returned by `model.frame` on the special handling of `NA`s.

In addition, non-null fits will have components `assign`, `effects` and (unless not requested) `qr` relating to the linear fit, for use by extractor functions such as `summary` and `effects`.

Using time series

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting `NA`s would invalidate the time series attributes, and if `NA`s are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or differenced regressor would be ignored. It is good practice to prepare a `data` argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and fitted values are time series.

Note

Offsets specified by `offset` will not be included in predictions by `predict.lm`, whereas those specified by an offset term in the formula will be.

Author(s)

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson & Rogers (1973).

References

Chambers, J. M. (1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G. N. and Rogers, C. E. (1973) Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, **22**, 392–9.

See Also

`summary.lm` for summaries and `anova.lm` for the ANOVA table; `aov` for a different interface.

The generic functions `coef`, `effects`, `residuals`, `fitted`, `vcov`.

`predict.lm` (via `predict`) for prediction, including confidence and prediction intervals; `confint` for confidence intervals of *parameters*.

`lm.influence` for regression diagnostics, and `glm` for **generalized** linear models.

The underlying low level functions, `lm.fit` for plain, and `lm.wfit` for weighted regression fitting.

More `lm()` examples are available e.g., in `anscombe`, `attitude`, `freeny`, `LifeCycleSavings`, `longley`, `stackloss`, `swiss`.

`biglm` in package **biglm** for an alternative way to fit linear models to large datasets (especially those with many cases).

Examples

```
require(graphics)

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

anova(lm.D9)
summary(lm.D90)

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

### less simple examples in "See Also" above
```

`glm` *Fitting Generalized Linear Models*

Description

`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)

## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

Arguments

`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`family`	a description of the error distribution and link function to be used in the model. For `glm` this can be a character string naming a family function, a family function or the result of a call to a family function. For `glm.fit` only the third option is supported. (See `family` for details of family functions.)
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `glm` is called.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`start`	starting values for the parameters in the linear predictor.
`etastart`	starting values for the linear predictor.
`mustart`	starting values for the vector of means.
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.
`control`	a list of parameters for controlling the fitting process. For `glm.fit` this is passed to `glm.control`.
`model`	a logical value indicating whether *model frame* should be included as a component of the returned value.
`method`	the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS): the alternative `"model.frame"` returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If specified as a character string it is looked up from within the **stats** namespace.
`x, y`	For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension `n * p`, and `y` is a vector of observations of length `n`.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`intercept`	logical. Should an intercept be included in the *null* model?
`object`	an object inheriting from class `"glm"`.
`type`	character, partial matching allowed. Type of weights to extract from the fitted model object. Can be abbreviated.
`...`	For `glm`: arguments to be used to form the default `control` argument if it is not supplied directly. For `weights`: further arguments passed to or from other methods.

Details

A typical predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. For `binomial` and `quasibinomial` families the response can also be specified as a `factor` (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with any duplicates removed.

A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first + second + first:second`.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms` object as the formula.

Non-`NULL` `weights` can be used to indicate that different observations have different dispersions (with the values in `weights` being inversely proportional to the dispersions); or equivalently, when the elements of `weights` are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations. For a binomial GLM prior weights are used to give the number of trials when the response is the proportion of successes: they would rarely be used for a Poisson GLM.

`glm.fit` is the workhorse function: it is not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

If more than one of `etastart`, `start` and `mustart` is specified, the first in the list will be used. It is often advisable to supply starting values for a `quasi` family, and also for families with unusual links such as `gaussian("log")`.

All of `weights`, `subset`, `offset`, `etastart` and `mustart` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

For the background to warning messages about 'fitted probabilities numerically 0 or 1 occurred' for binomial GLMs, see Venables & Ripley (2002, pp. 197–8).

Value

`glm` returns an object of class inheriting from `"glm"` which inherits from the class `"lm"`. See later in this section. If a non-standard `method` is used, the object will also inherit from the class (if any) returned by that function.

The function `summary` (i.e., `summary.glm`) can be used to obtain or print a summary of the results and the function `anova` (i.e., `anova.glm`) to produce an analysis of variance table.

The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` can be used to extract various useful features of the value returned by `glm`.

`weights` extracts a vector of weights, one for each case in the fit (after subsetting and `na.action`).

An object of class `"glm"` is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`residuals`	the *working* residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are `NA`.
`fitted.values`	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
`rank`	the numeric rank of the fitted linear model.
`family`	the `family` object used.
`linear.predictors`	the linear fit on link scale.
`deviance`	up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
`aic`	A version of Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the `aic` component of the family. For binomial and Poison families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. For families fitted by quasi-likelihood the value is `NA`.
`null.deviance`	The deviance for the null model, comparable with `deviance`. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
`iter`	the number of iterations of IWLS used.
`weights`	the *working* weights, that is the weights in the final iteration of the IWLS fit.
`prior.weights`	the weights initially supplied, a vector of `1`s if none were.
`df.residual`	the residual degrees of freedom.
`df.null`	the residual degrees of freedom for the null model.
`y`	if requested (the default) the `y` vector used. (It is a vector even for a binomial model.)
`x`	if requested, the model matrix.
`model`	if requested (the default), the model frame.
`converged`	logical. Was the IWLS algorithm judged to have converged?
`boundary`	logical. Is the fitted value on the boundary of the attainable values?
`call`	the matched call.
`formula`	the formula supplied.
`terms`	the `terms` object used.
`data`	the `data` argument.
`offset`	the offset vector used.
`control`	the value of the `control` argument used.
`method`	the name of the fitter function used, currently always `"glm.fit"`.
`contrasts`	(where relevant) the contrasts used.
`xlevels`	(where relevant) a record of the levels of the factors used in fitting.
`na.action`	(where relevant) information returned by `model.frame` on the special handling of `NA`s.

In addition, non-empty fits will have components `qr`, `R` and `effects` relating to the final weighted linear fit.

Objects of class `"glm"` are normally of class `c("glm", "lm")`, that is inherit from class `"lm"`, and well-designed methods for class `"lm"` will be applied to the weighted linear model at the final iteration of IWLS. However, care is needed, as extractor functions for class `"glm"` such as `residuals` and `weights` do **not** just pick out the component of the fit with the same name.

If a `binomial` `glm` model was specified by giving a two-column response, the weights returned by `prior.weights` are the total numbers of cases (factored by the supplied case weights) and the component `y` of the result is the proportion of successes.

Fitting functions

The argument `method` serves two purposes. One is to allow the model frame to be recreated with no fitting. The other is to allow the default fitting function `glm.fit` to be replaced by a function which takes the same arguments and uses a different fitting algorithm. If `glm.fit` is supplied as a character string it is used to search for a function of that name, starting in the **stats** namespace.

The class of the object return by the fitter (if any) will be prepended to the class returned by `glm`.

Author(s)

The original R implementation of `glm` was written by Simon Davies working for Ross Ihaka at the University of Auckland, but has since been extensively re-written by members of the R Core team.

The design was inspired by the S function of the same name described in Hastie & Pregibon (1992).

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

`anova.glm`, `summary.glm`, etc. for `glm` methods, and the generic functions `anova`, `summary`, `effects`, `fitted.values`, and `residuals`.

`lm` for non-generalized *linear* models (which SAS calls GLMs, for 'general' linear models).

`loglin` and `loglm` (package **MASS**) for fitting log-linear models (which binomial and Poisson GLMs are) to contingency tables.

`bigglm` in package **biglm** for an alternative way to fit GLMs to large datasets (especially those with many cases).

`esoph`, `infert` and `predict.glm` have examples of fitting binomial glms.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
anova(glm.D93)
summary(glm.D93)

## an example with offsets from Venables & Ripley (2002, p.189)
utils::data(anorexia, package = "MASS")

anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
                family = gaussian, data = anorexia)
summary(anorex.1)


# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))
summary(glm(lot1 ~ log(u), data = clotting, family = Gamma))
summary(glm(lot2 ~ log(u), data = clotting, family = Gamma))

## Not run:
## for an example of the use of a terms object as a formula
demo(glm.vr)

## End(Not run)
```